ARMY RESEARCH LABORATORY

Gains in the Education of Mathematics and Science GEMS: Teaching Robotics to High School Students

by Edward M. Measure and Edward Creegan

ARL-TR-6220 **January 2013**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
White Sands Missile Range, NM 88002-

ARL-TR-6220 **January 2013**

Gains in the Education of Mathematics and Science (GEMS): Teaching Robotics to High School Students

Edward M. Measure and Edward Creegan
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2013	Final	

4. TITLE AND SUBTITLE

Gains in the Education of Mathematics and Science (GEMS): Teaching Robotics to High School Students

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Edward M. Measure and Edward Creegan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: RDRL-CIE-D
White Sands Missile Range, NM 88002-

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Since 2008, the U.S. Army Research Laboratory (ARL) components at White Sands Missile Range in New Mexico, consisting of elements of the Computer and Information Sciences Directorate (CISD) and the Survivability, Lethality, and Analysis Directorate (SLAD), have conducted summer Gains in the Education of Mathematics and Science (GEMS) programs for students from high schools in the major towns nearby.

The several modules of this program teach students how math and science relate to the Army, ARL, and in particular, about technical areas in which we are involved. Since the beginning of the program, the authors have taught two of these modules on robotics. In addition to teaching something about the history, evolution, and current status of robotics, we use robotics kits to allow students to build, program, and test robots that can use electronic sensors and student-built computer programs to detect and maneuver through obstacles. In this report we include descriptions and pictures of class activities and detailed class materials.

15. SUBJECT

Robotic, GEMS, Education

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Edward M. Measure
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	62	19b. TELEPHONE NUMBER (*Include area code*) (575) 678-3307

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

Preface

The U.S. Army's Gains in the Education of Mathematics and Science (GEMS) program is an educational outreach program intended to expose selected high school and middle school students to mathematics, science, the Army, and its research programs. The U.S. Army Research Laboratory (ARL) has participated in this program, and the authors have been part of the program at White Sands Missile Range (WSMR) in NM. The WSMR program has consisted of a series of modules, presented four per day over the course of a week to groups of eight to ten students in each. Each module presenter teaches four groups per day.

For the past several years, we have taught two course modules on robotics, titled Robotics I and II, to about 200 high school students as part of the GEMS program. The brevity of each module, just 75 min, puts a real premium on concision and organization. This report documents our experiences and some lessons we have learned.

A very large number of individuals have contributed to the success of the ARL GEMS program at WSMR, including a changing cast of organizers; module instructors; and those who supported the rather complex logistics of transporting, feeding, and entertaining students on a closed post fairly far from their homes. We are immensely grateful to Chris Rodriguez, Tom Maxwell, Kurt Austin, Butch Peel, Terry Jameson, Rudy Velasquez, Gina Selga, and Lori Hungate-Diehl, who have been particularly helpful to us, and especially those who worked more anonymously behind the scenes to permit students to show up in our classroom, ready to learn. Including, but not limited to, Joseph JoJola, Charles Perez, Don Hoock, MAJ Bateman, SGT Huff, Jeremy Gonzalez, Elliot Bergsagel, Rudy Montoya, and Johnny Infante.

Executive Summary

Since 2008, the U.S. Army Research Laboratory (ARL) components at White Sands Missile Range in New Mexico, consisting of elements of the Computer and Information Sciences Directorate (CISD) and the Survivability, Lethality, and Analysis Directorate (SLAD), have conducted summer Gains in the Education of Mathematics and Science GEMS programs for students from high schools in the major towns nearby.

The several modules of this program teach students how math and science relate to the Army, ARL, and in particular, about technical areas in which we are involved. Since the beginning of the program, the authors have taught two of these modules on robotics. In addition to teaching something about the history, evolution, and current status of robotics, we use robotics kits to allow students to build, program, and test robots that can use electronic senses and student-built computer programs to detect and maneuver through obstacles. In this report we include descriptions and pictures of class activities and detailed class materials.

INTENTIONALLY LEFT BLANK.

1. Introduction: Background of the Program

Because White Sands Missile Range (WSMR) is an isolated and closed Army post, simply getting the Gains in the Education of Mathematics and Science (GEMS) students here is a challenge. Most of our students come from either Las Cruces, NM, which is 26 miles away, or El Paso, TX, which is twice that far. In order to participate, they need to get up early on a summer's day, catch a bus, and endure a fairly long ride to get here by 8:00 a.m. The logistics are arranged by U.S. Army Research Laboratory (ARL), but students still need to rise bright and early, or at least early, and put in half a day before many of their classmates have gotten out of bed.

Our recent practice has been to run two 1-week sessions each summer, one for Las Cruces students and one for those from El Paso. Opportunities to participate are advertised in local high schools, and application can be made on the Web, where students fill out a questionnaire and discuss why they would like to participate. Our classes have room for about forty students, in four teams of ten each, so a selection process winnows the selectees down to that number.

In return, they get lunch, some entertainment, a lot of instruction in diverse technologies, exposure to real Soldiers and some of their jobs, and a stipend if they complete the full week. Scientists and engineers can hardly be trusted to supervise a bunch of high school students, of course, so the program also recruits teachers, who also get a stipend.

Students are organized into teams of eight to ten; each led by a teacher, and usually go from module to module with the other members of their team. Thus each ARL module teacher has one small class at a time with a high school teacher there for support.

Our communities are mostly Hispanic but still quite culturally diverse, and our students are also diverse in age, educational experience to date, and career goals. Perhaps 20% of our students already intend to pursue a military career and are anxious to get a head start on it. A somewhat larger group is already planning a career in science or technology and the rest have other goals or are undecided. As a result of this diversity, we cannot count on too much scientific or mathematical preparation.

2. Plan of the Course

2.1 Class Materials

Our biggest challenge was to understand how we could pack a meaningful exposure to a vast field into two short 75-min sessions, and our biggest advantages were the facts that most students have been extensively exposed to many concepts through news and movies and find robots

intrinsically interesting. We knew that we wanted the course to have a strong hands-on element while also giving the students some exposure to the history and theory of the subject.

The necessity of a hands-on element meant that we wanted students to have some experience building and programming an actual robot. Cost, flexibility, and programmability were our primary criteria. We considered several varieties of kit type robots and settled on the LEGO® MINDSTORMS®. The building blocks of this kit consist of the usual LEGO® snap-together parts, sensor and motor units that snap into the other LEGO® parts, a controller unit, and a graphic and highly intuitive high-level programming language based on National Instruments LabVIEW (National Instruments, 2012). LabVIEW proper has become a very popular program for engineers and scientists putting together measurement systems and sensors. Programs for the NXT control unit are constructed using a drag and drop interface. These kits have great flexibility and permit a good deal of functionality.

One of the designs that can be built with the LEGO® MINDSTORMS® kit is called *The Explorer*, and it exhibits what we consider the three key attributes of a modern robot: movement, sensing of the environment, and decision making and action based on that sensor input. These attributes permit and require programming to allow the robot to respond sensibly to its environment. Programming of the controller units is best done on a separate computer

2.2 Class Organization

The typical student team consists of ten students, a teacher, and, sometimes, a student helper who is a veteran of a previous course. We organize the students in pairs, give each pair a LEGO® MINDSTORMS® kit and a laptop computer for programming, and attempt to ensure that each student tries out each aspect of assembly and programming.

We think there would be both educational advantages and disadvantages to having each student work individually, but considerations of time and cost were the primary determinants of the pair-based structure. Teaming sacrifices some individual learning; however, it is good preparation for actual work in engineering, where nearly all work is teamwork.

There are five major components to our instruction: lecture, media, construction, programming, and experimentation. Our first module, Robotics I, begins with a history of robotics (see detailed lecture slides in appendix A). Here we emphasize how robotic represent a confluence of the eighteenth and nineteenth century mechanical technologies with the electronic sensing and computing technologies of the twentieth, and how modern robotics has grown explosively, driven mainly by industrial and military applications, but with consumer products increasingly entering the picture.

2.3 Class Lessons

We want our students to have the experience of assembling their robots, but we found that assembly from scratch was just too time consuming. Consequently, we have adopted the

stratagem of giving them partially assembled units, with many of the more tedious steps already done, plus presorted parts for additional stages of construction.

The most important advantage of this approach is that it permits us to begin the crucial task of teaching them how to program at an early stage. Working with their partially preassembled units allows them to quickly complete the first stage construction, which results in a proto-robot with movement capability but no sensors, except those intrinsic to the servo-motors, which keep track of their rotation state and history. At that point we have them bring up their programming interface and show them how to use the drag, drop, and specify parameters interface to cause their proto-bots (a yet incompleted robot) to go forward, backward, and turn. Each programming stage is introduced with a short talk on the programming concepts which is immediately followed by implementation and experimentation. We also show them how to use the programming language's loop structure to permit repetitive sequences of motions.

The remainder of the module is spent allowing them to experiment with programming and testing their not yet fully functional robots.

The second module, imaginatively styled Robotics II, again begins with a lecture and media presentation, this time focused on videos of modern robots, with emphasis on using concepts adapted from living creatures, or biomimetics, which is a major focus of our own work. (See appendices B and C on class materials for details).

Next, we put them back into robot assembly, incorporating a touch sensor in their designs. The students retrieve their partially assembled robots, which have been labeled and saved from the previous module. When the next stage of assembly is complete they are ready to learn and incorporate another programming concept—using the input from the touch sensor to control a decision about what the robot should do next. This is a simple but popular activity, and soon the floor is alive, or at least mechanically active, with little robots running into walls, chairs, people, or each other, deciding what to do next and then continuing on their way.

The final construction stage incorporates a rotatable ultrasonic distance sensor. Achieving the full capability of the *Explorer* requires a significantly more intricate computer program, but at this point the students have already had experience with the major programming concepts. Refinements added at this stage are logical and numerical decision blocks and control lines (see appendix C on programs).

Testing, debugging, and experimentation complete the class. The completed *Explorer* is pretty capable. It proceeds until it encounters an obstacle, either by touch or via its ultrasonic sensors. If there is an obstacle ahead, it uses its ultrasonic sensor to look left and right and turns itself and proceeds in the direction that is clearest. Students are invariably impressed with the way their robots maneuver through obstacles consisting of walls, doorways, and a forest of chairs, tables, and human legs. We also have magnetic and light sensors for the robots which could be used for goal seeking activity, but we have never managed to find enough class time to add that

functionality. If we were ever to add a third robotics module, that might be an important component.

3. Robotics in Action

Mr. Jojola photographed the GEMS students and their instructors in action; some of the photos of our robotics classes are included.

First we talk about the history and status of robotics, as well as why we are interested in the subject. Interest tends to perk up when we point out the stack of kit boxes in the back of the room. At this stage, we try to connect the real history of robotics and the imaginary parts featured in movies, as well as try to separate fact from fiction (figure 1).

Figure 1. So when do we get to open up the toy boxes?

After the introduction of the history of robotics, we instruct the students on how we work on the kits. Students work in teams of two, and each team keeps its kit for the duration of the 2 days of instruction, so kits need to be labeled accordingly (figure 2).

Figure 2. Before you get to play, you have to listen to us talk, OK. Ed Creegan is relating the history of robotics.

Many or most students have worked with Legos as children or even recently. We try to arrange the pairs so that at least one of the students has some prior Lego experience, but we also want to avoid permitting one to watch while the other does all the work. In figure 3, bottom center, our partially assembled demo robot is available for the students to check out if they cannot quite figure out what goes where. We also walk around and try to solve any problems that come up, despite careful preparation, missing parts are common, but we keep an extra supply.

These students in figure 3 are assembling the robots from scratch; however, we later decided to give them kits with partially assembled robots, so that they would have more time to work on programming. Programming, however, is not very photogenic.

Figure 3. Begin construction.

In figure 4, our students are just beginning to assemble the robot and are picking out the parts needed for the robot's undercarriage which will mount motors and wheels.

Figure 4. These 1 × n pieces all look alike. You need to count the holes to figure out which is which.

Sensors and motors are connected to the control unit by cables. We have two modes for connecting the laptops, where programs are written, to the controllers: USB hardwire and Bluetooth. When these photos were shot, we didn't yet have permission to use the Bluetooth systems.

In figure 5, our student is preparing to download the program pictured on her laptop screen to her robot and is in the process of connecting a USB cable from laptop to robot controller.

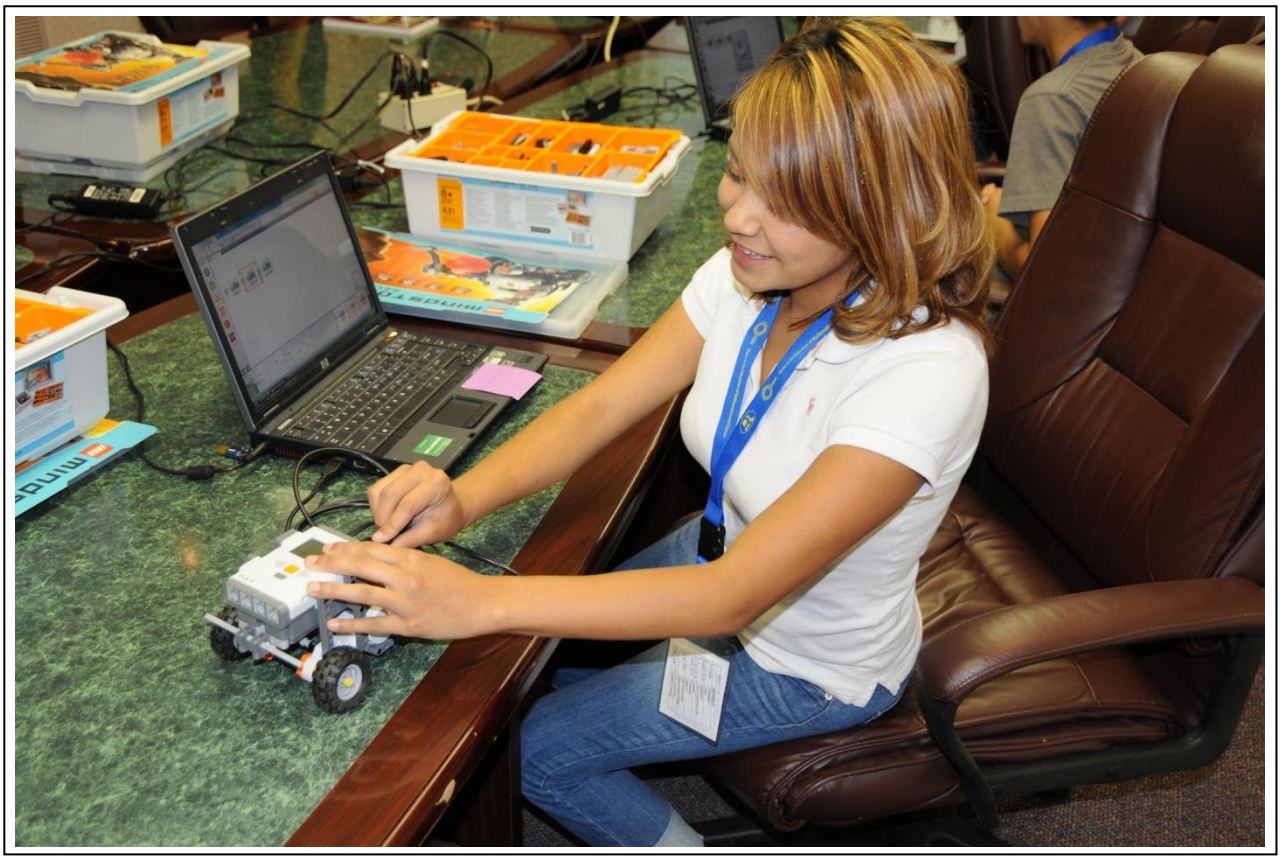


Figure 5. OK, so we connect the USB port of the laptop to that of the controller.

Sometimes the robots won't go if we forget to tell the students how to turn them on. In figure 6, Ed Measure is showing this pair of students the ON button.

Figure 6. I think I forgot to mention that you need to push this orange button to turn the robot on.

In figure 7, the student in the foreground has completed assembly of the mast mounted ultrasonic sensor unit and is giving it a final inspection. These mast mounted units can be steered to point in any direction using the motor unit connect to the A portal.

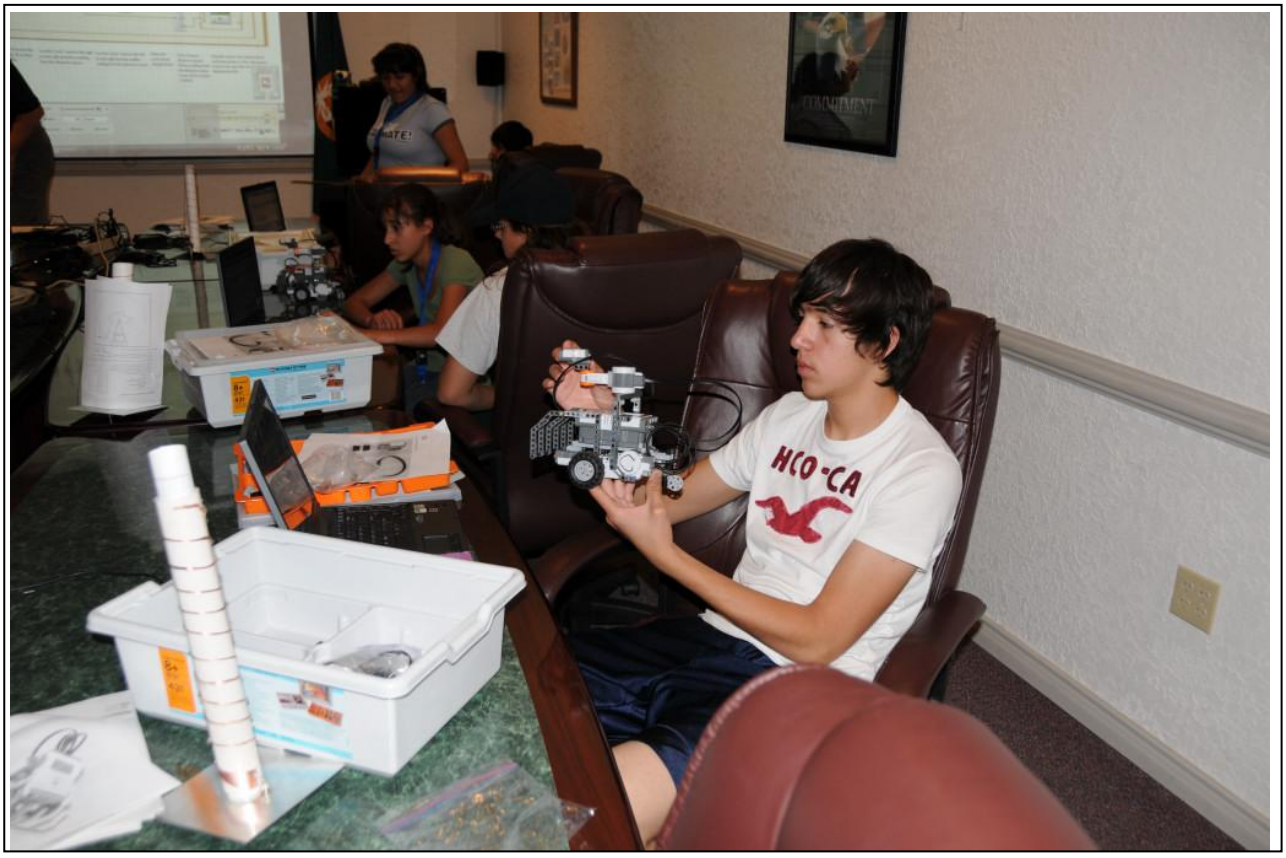


Figure 7. Prepare to test ultrasonic sensor.

In figure 8, a misbehaving robot has been partially disassembled to see why it doesn't want to go. Either mechanical problems or programming problems might be the culprit here.

Figure 8. Sometimes debugging requires partial disassembly. Robot knee bone connected to robot shin bone? Check. Maybe it's a programming problem? Hmm?

Download and test are underway in figure 9. Progress is facilitated by a very short path between program building and testing.

Figure 9. Downloading program from laptop to robot.

Great expectations, near misses, observations, and escaping robots are seen in figures 10–15

Figure 10. OK, Mr. Roboto, let's see what you've got.

Figure 11. Uh oh, it's headed for the edge. Come back you rascally robot!

Once the programmed robots start doing their own decisions and maneuvers, it's very easy to start thinking anthropomorphically. The robots in figure 12 have just gotten snagged on each other, but it sure looks like they just stopped to have a conversation.

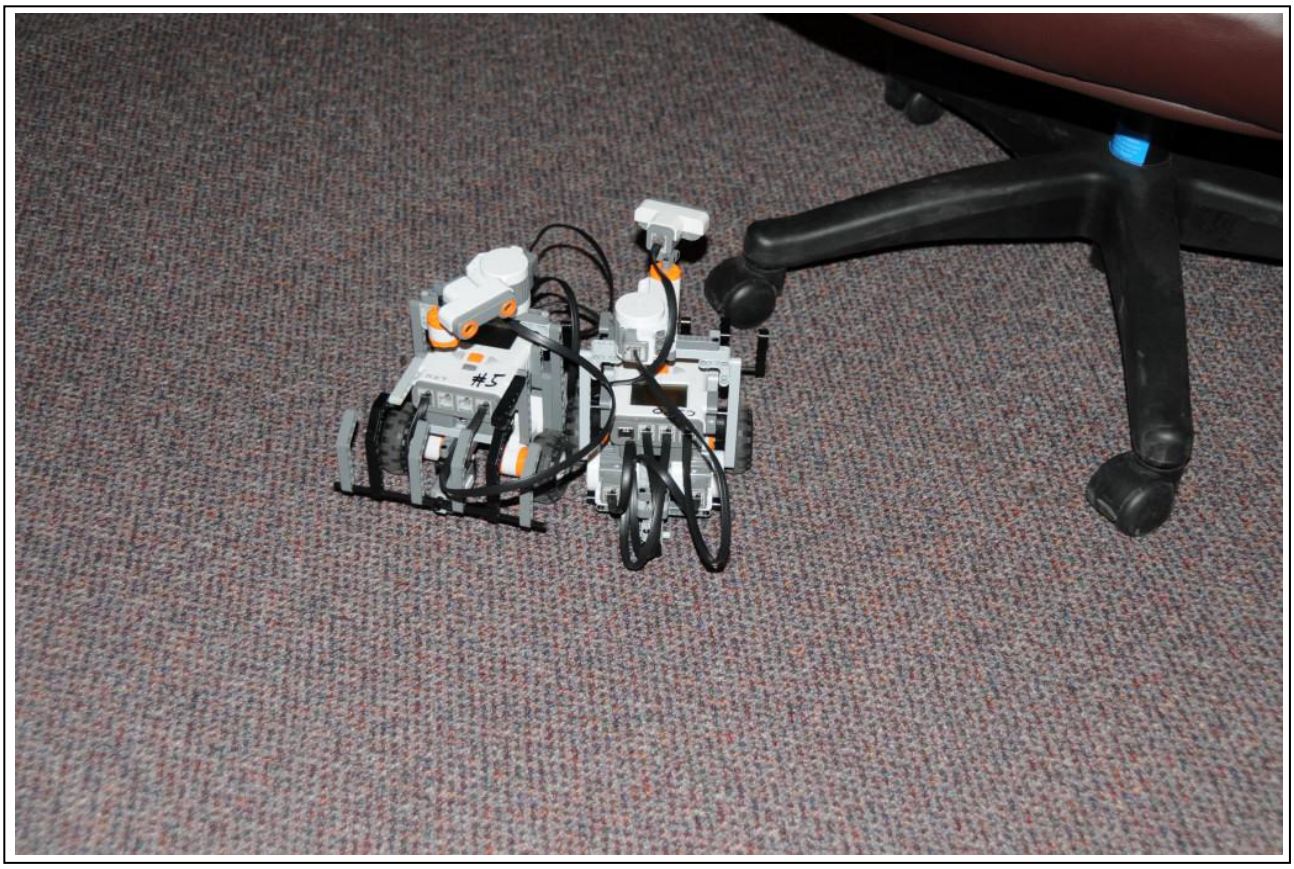


Figure 12. How do things look your way Number 5? I sense some obstruction in my direction.

Figure 13. OK, obstruction ahead. I hate to think of the size of whatever it is attached to those shoes.

Figure 14. Check out the doorway. You go right on a decoy route, and I'll make a break for it!

Figure 15. Cut him off! Cut him off! He's making a getaway.

4. Conclusions, Lessons Learned, and Future Prospects

The ARL WSMR GEMS program receives feedback from the students and teachers at the end of the week, and occasionally we receive an out of cycle accolade. One that thrilled all of us responsible for the program was a letter from a teacher who wrote that she felt the program had helped a student turn his life around and become a serious, goal directed student rather than just another cut up. Such an outcome is a best case scenario, we suppose, but most of what we hear from students is very positive. They like the learning style and enjoy learning about the Army and some of its technical activities.

Ideally, we would like to get some of the students interested in the science and technology of robots. We also want them to know that science has a lot of connections to other sciences and engineering disciplines, especially computer science, electrical and mechanical engineering, physics and mathematics, but also, and increasingly, to biology and perhaps even sociology. Mathematics is at the core of most of the above, so we emphasize that those interested in robotics really should master as much mathematics as they can—advice we think is pretty good even if they do not plan to study robotics.

Most of the lessons we have learned are about what not to do. In particular, we learned that time is a very precious commodity and we need to ensure that we do not spend too much time on any specific aspect. Our biggest adaptations in that respect have been in cutting down on our lecture time (we love to talk robots and could talk all day). Another time saving adaptation is to start with partially constructed robots so that phase of the module will not be a roadblock to other progress. We also have found that the students are themselves pretty good at finding ways to waste time, so we have stopped teaching them some tricks that they find amusing but that we find of less educational value, like having the robots say comical things.

Those who have more teaching time would doubtless make other choices. In particular, the robots can be taught more complex behaviors; some types of goal seeking behaviors are a logical next step. More complicated programming techniques have their place, but the additional investment of time required is large, probably many times our total class time.

Naturally, we have some frustrations with the limitation of the MINDSTORMS® platform itself. Constructions lack durability and the unreasonably limited memory of the control module means that much longer and more complex programs are not practical.

More time for open ended exploration would be advantageous; students choose and pursue their own design, construction, and programming goals. Those must be reserved for those who have a lot more time.

5. References and Bibliography

LEGO® MINDSTORMS®, 2012: http://mindstorms.lego.com/en-us/products/default.aspx, accessed January 26, 2012.

LEGO® MINDSTORMS®, 2012a: http://mindstorms.lego.com/en-us/whatisnxt/default.aspx, accessed January 26, 2012.

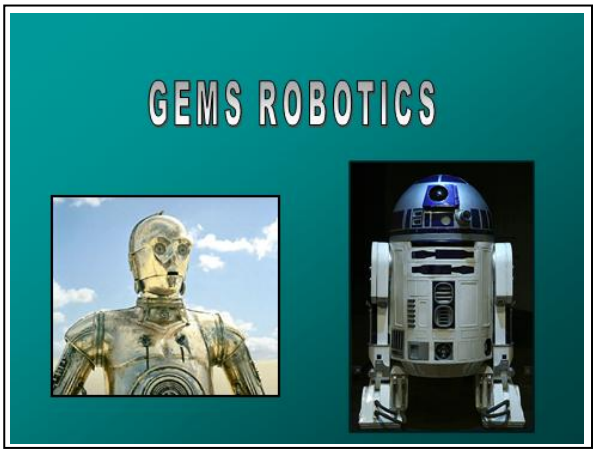
National Instruments, 2012: http://www.ni.com/labview/, accessed January 26, 2012.

Appendices

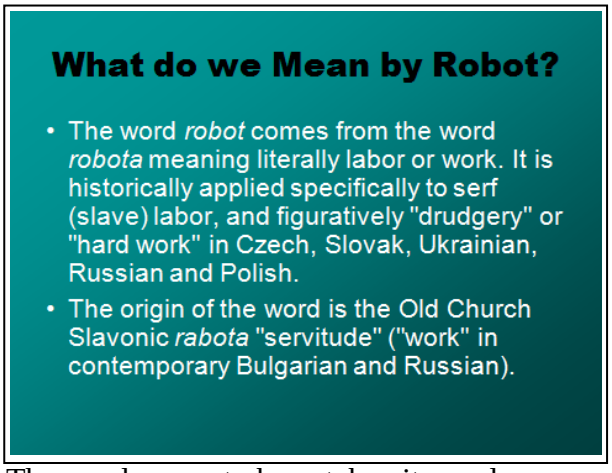
Because we wanted this report to be extensive enough to allow others to copy our lessons, we have included details about the slides and videos we show and the programming concepts that we teach. Of course, we adjust the content of the course from year to year and others are free to choose different paths, but we hope that including these details might be helpful to someone starting a similar program. The three following appendices detail our lecture slides and some related notes, our videos, and our program examples respectively.

Appendix A. Slides and Lecture Notes

Our slides, with some commentary generally similar in character to what we say in class.



Our generation got a lot of its introduction to robotics from R2D2 and C3PO, the personable robots from *Star Wars.* Of course the concept is a lot older.



The word seems to have taken its modern meaning from a play called *Rossum's Universal Robots*, by Czech playwright Karel Čapek, who attributed invention of the word to his brother Josef. The play was about a factory populated by sentient androids.

What do we Say Now?

- The Robotics Institute of America (RIA) defines a robot as:

  A re-programmable multi-functional manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.
- The RIA recognizes four classes of robot:

  1: Handling devices with manual control

  2: Automated handling devices with predetermined cycles

  3: Programmable, servo-controlled robots with continuous of point-to-point trajectories

  4: Robots capable of Type C specifications which also acquire information from the environment for intelligent motion

Well maybe. To me, number one sounds like a screwdriver, and number two could, well, be my toaster. Number three is getting closer, but I don't really call it a robot until we get to number four. It's the combination of sensing the environment and reacting to it that makes a real robot, and that's the kind we build in our class.

What we Really Think

- Joseph Engelberger, a pioneer in industrial robotics, once remarked: "I can't define a robot, but I know one when I see one."

Well, maybe, but it sounds pretty evasive to me. I will stick with the sensing plus reacting definition.

Start of Modern Robots

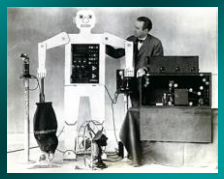
- Military Systems drive research
- In 1898 Nikola Tesla publicly demonstrated a radio-controlled (teleoperated) boat, similar to a modern radio operated vehicle. Tesla hoped to develop the wireless torpedo into a weapon system for the US Navy.

The military is a natural place for robotics, since one of the big advantages is that a robot can go in harm's way without endangering a person. The first attempts at robotics were teleoperated devices – systems remotely controlled but without necessarily having their own on board sensors or automated controllers. These would not be considered class four robots, or necessarily even class three robots, but they continue to be important steps in the direction of true robotics

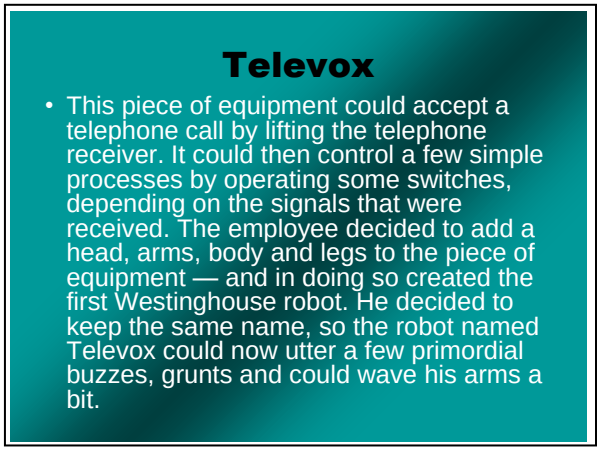
Most of the so-called drones flown by our military, more technically called unmanned aerial systems, or UAS, are in this category. They are, however, in the process of incorporating more and more on-board control, making them more nearly autonomous.

Make Life Easy!

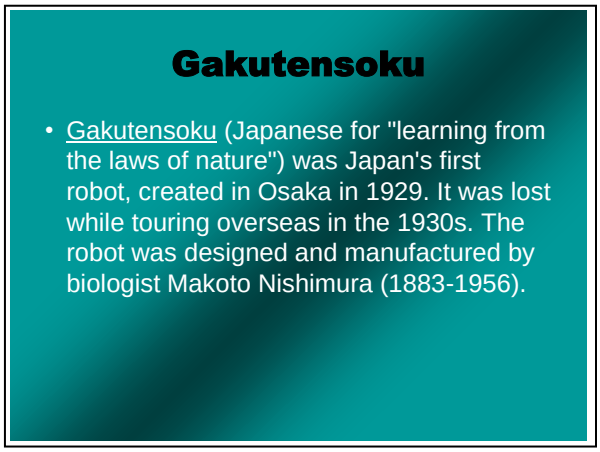
- In 1926, Westinghouse Electric Corporation created Televox, the first robot put to useful work.



The ideas of robotics captivated technologists long before they learned how to make them do anything very useful. The Westinghouse Televox was pretty much limited to picking up a telephone, but with a few humanoid features, it looked more likeable.



Cute, but it doesn't seem to have revolutionized the world. The world was still taking baby steps toward robotics, mainly because like the Scarecrow in *The Wizard of Oz*, the robot still lacked a brain. Remedying that lack would have to wait on the invention of the computer.



Not much seems to be known about Nishimura's robot, but the idea of imitating biological systems has been a powerful one in robotics. Of course the original idea in robotics was to imitate people, but the idea of imitating other animals turns out to be a good one too.

Electronic Autonomous Robots

- Were first created by William Grey Walter of the Burden Neurological Institute at Bristol, England in 1948 and 1949. They were named *Elmer* and *Elsie*. These robots could sense light and contact with external objects, and use these stimuli to navigate.

For a full robot capability, it needs to be able to operate independently of a human operator, at least in some respects. That independent kind of operation, or autonomy, depends on having its own ways to sense the environment and react to the information coming from that environment. The invention of the transistor just before the middle of the twentieth century made possible small and powerful electronic sensors and computers that permitted autonomous capabilities to be developed.

Our *Explorer* incorporates such sensors and such a computer.

Robotics competitions have become a popular activity for students and others. The robot pictured is designed to follow the black line on the floor and to deposit a ball in the box in front of it.

These competitions test and develop skill in design, mechanical construction, and programming.

Put Them to Work

- The first truly modern robot, digitally operated, programmable, and teachable, was invented by George Devol in 1954 and was ultimately called the Unimate. It is worth noting that not a single patent was cited against his original robotics patent (U.S. Patent 2,988,237). The first Unimate was personally sold by Devol to General Motors in 1960 and installed in 1961 in a plant in Trenton, New Jersey to lift hot pieces of metal from a die casting machine and stack them.

The best thing about robots is that they can be good at jobs that are difficult, dangerous, boring, and otherwise unpleasant for people.

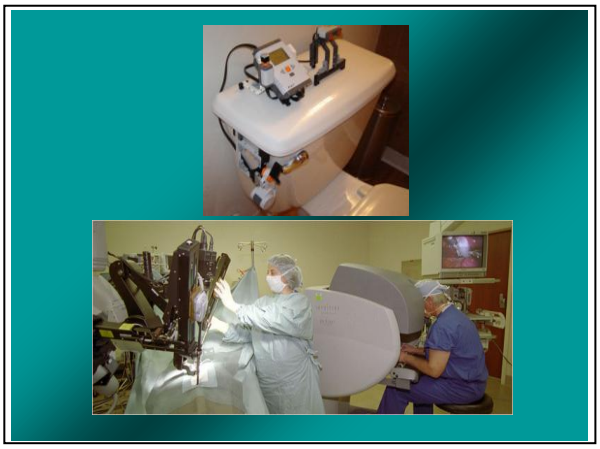
The gigantic robots on the right are assembling huge trucks on a long assembly line. The little guy on the left, a Roomba, is a household robot that can autonomously vacuum a room while driving your cat to distraction.

Robots have become ubiquitous in the factory and in ordinary life, though we sometimes don't recognize them as such. Your robotic digital video recorder will search the television listings for your favorite programs and record them even when you are away. A robotic cop will spot you speeding or going through a red light, take your picture, and send you a ticket.

Yeah, they're COOL, but why build robots?

- **Dirty, dangerous, dull or inaccessible tasks**
- Too boring to bother with, for example domestic cleaning
- Too dangerous, for example exploring inside a volcano.
- Physically inaccessible. For example, exploring another planet, cleaning the inside of a long pipe or performing laparoscopic surgery.
- News Flash: Some jobs they DO BETTER!!

Of course that's just one side of the story. The other side is that they are not just able to jobs we don't like to do, but they can also do a lot of jobs more cheaply than people can, and they are busy taking those jobs as well. Like any other technology, robotics has a downside.



One job robots are already doing in airports and some other businesses is flush toilets for those too lazy or careless to do so. The *Mindstorms* design at the top performs the same function. Below, a surgeon at right performs a robotically assisted surgery with the aid of the machines and nurses at right. Robotic assist can still any trembling in the surgeons hands and tiny robot "fingers" can go in places where a surgeons hands would never fit. Surgery might even be performed by a physician thousands of miles away.

Dangers and Fears

- Robots could be dangerous if they were programmed to kill or if they are programmed to be so smart that they make their own software, build their own hardware to upgrade themselves or if they change their own source code. *The Terminator, Runaway, Robocop*, *Stargate*, the Cylons in *BattleStar Galactica*, The Matrix, and *I, Robot*.

So how are we doing on these criteria? Our war robots are mostly under direct human control at the moment, but that's probably just temporary. Fully autonomous fighter jets are being developed and will probably be needed, since the human brain is just too slow to compete with an electronic one. Computers already play a central role in the design and building of robots, and that role is likely to expand. Computers do a certain amount of programming as well.

The Future?

At the present, robotic technology is being driven more by defense needs than any other factor. Unmanned Aerial Systems (UAS), popularly known as drones, have become one of the most important parts of our airborne forces. At first they served purely for surveillance, but they were quickly weaponized and now play a crucial role in global force projection.

The iRobot Warrior, here looking armed and dangerous, is a new robotic platform from the same people who brought you the Roomba vacuum cleaner. The Warrior is not just a fighting platform. When equipped with its manipulator arm, it's strong enough to tow a truck and dexterous enough to open a trunk.

Appendix B. Videos

Figure B-1. Who says this ant can't dance?

Video at: http://youtu.be/GDaNkff5Yyg

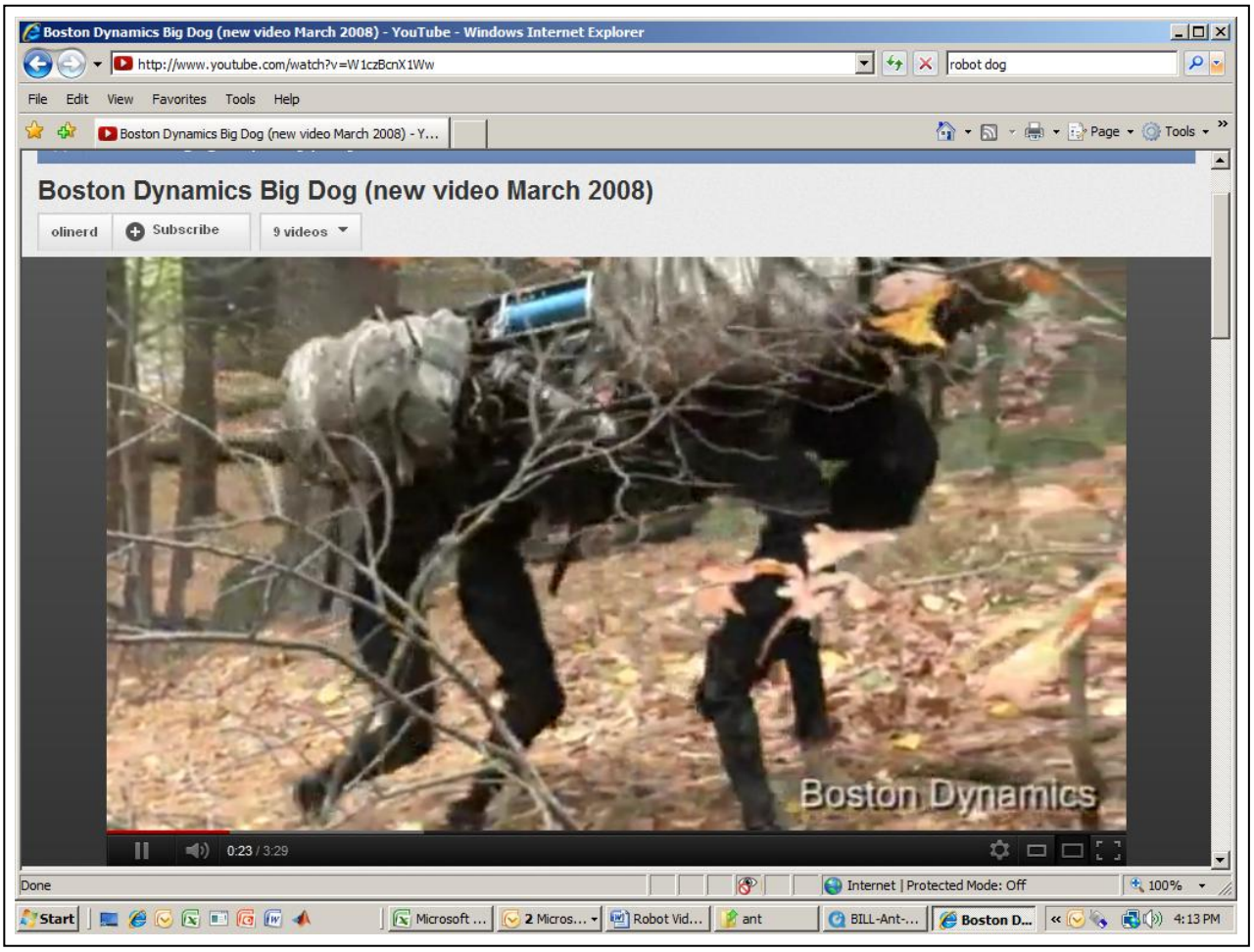


Figure B-2. The Boston Dynamics Robot Dog.

Video at: http://youtu.be/W1czBcnX1Ww

Figure B-3. A robot of the whegs family in action.

Video at: http://youtu.be/pNi2ytOdbTY

Figure B-4. Dragon Runner: a highly durable battlefield throwbot.

Video at: http://youtu.be/GmPMlT6XpHM

Figure B-5. University of Pennsylvania quadcopter.

Video at: http://youtu.be/E7X0_6o9J10

Daniel Mellinger of Vijay Kumar's lab at University of Pennsylvania has posted a number of videos demonstrating the aerial feats of his quad rotor helicopters. The video describes the copters as autonomous, but that's an exaggeration, as they get position input from twenty very high speed cameras. The precision of the maneuvers is impressive, however.

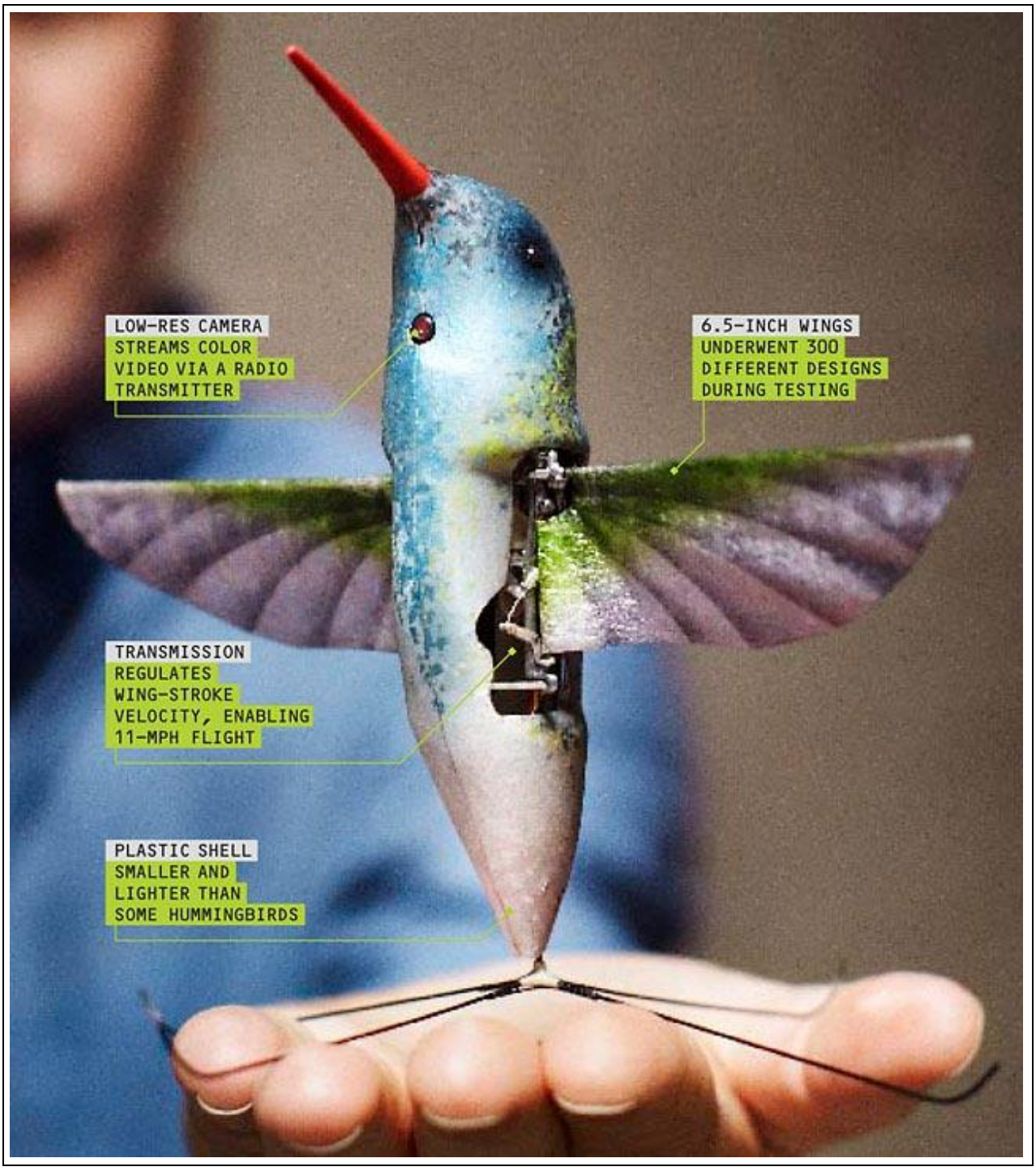


Figure B-6. The DARPA/AeroVironment nano air vehicle.

Video at: http://youtu.be/a8ZbtZqH6Io

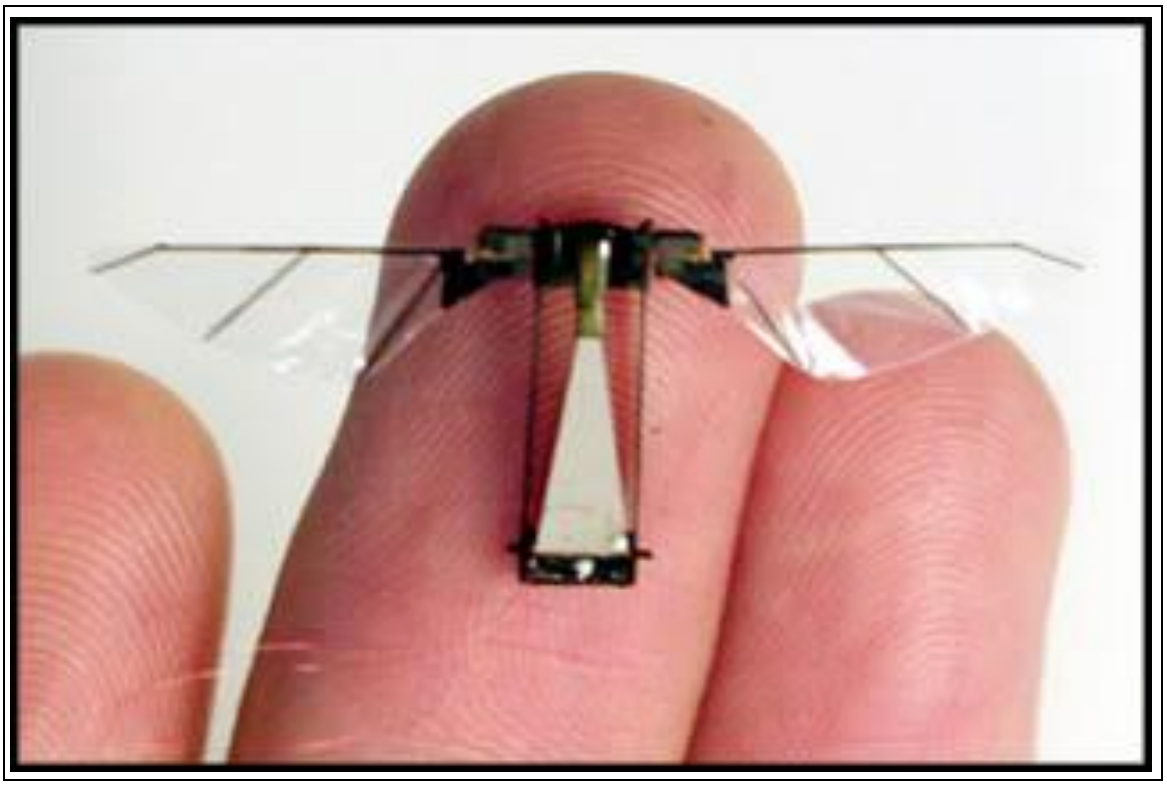


Figure B-7. Robert Wood's Harvard robot fly.

Video at: http://youtu.be/fFpov-ZSujA

Now this is what I would call a nano air vehicle – only about 1/50 of the mass of the DARPA system. This is a very small flying device, but it doesn't yet have sensors, power, or real control on board yet – so we have a ways to go before we can compete with nature at this scale.

INTENTIONALLY LEFT BLANK.

Appendix C. Programs and Discussion

One of the most crucial parts of our program is the introduction to robotic programming. As with every aspect of a program allotted only 150 min, that instruction needs to be very compressed. Our approach is to present some very simple example programs, have the students recreate them on their computers, and test them out with their robots. We write the programs on a laptop that displays on a projection screen and the students imitate on their laptops. Our experience is that copy errors are learning opportunities! One of the most valuable programming lessons is seeing what happens when you did it wrong.

The initial programs require only the most elementary capabilities of their robots – movement only, without sensing. A crucial part of the instruction is to let students experiment – play – with their programs and try different possibilities at each stage. We start with the empty program and show how to build programs from the graphic elements found on the left side of the image below, figure C-1. That leftmost column of icons is a menu of program elements from which programs can be built. There are actually three such menus, selected by the three small icons at the bottom of the icon column. The left most is the so-called simple menu, and it is the one we usually display in the figures below, since it is slightly more intuitive. The middle menu is called the complete menu, and its blocks open to display submenus which allow access to more complex programming concepts. Finally, the right most menu choice is a custom menu, which can be equipped with customized blocks. We don't use it.

We are mostly concerned with five types of program blocks, motor blocks, sensor blocks, loop blocks, switch blocks, and logic blocks. The motor block is exemplified on the menu by the top icon, a block with two gears on it.

This image below shows the start screen for the NXT programming development system. The icons on the left hand side represent categories of program objects that can be dragged to and dropped in the light blue box in the left center of the gridded area. The top block is a motor block, while the bottom two are loop and switch constructs respectively.

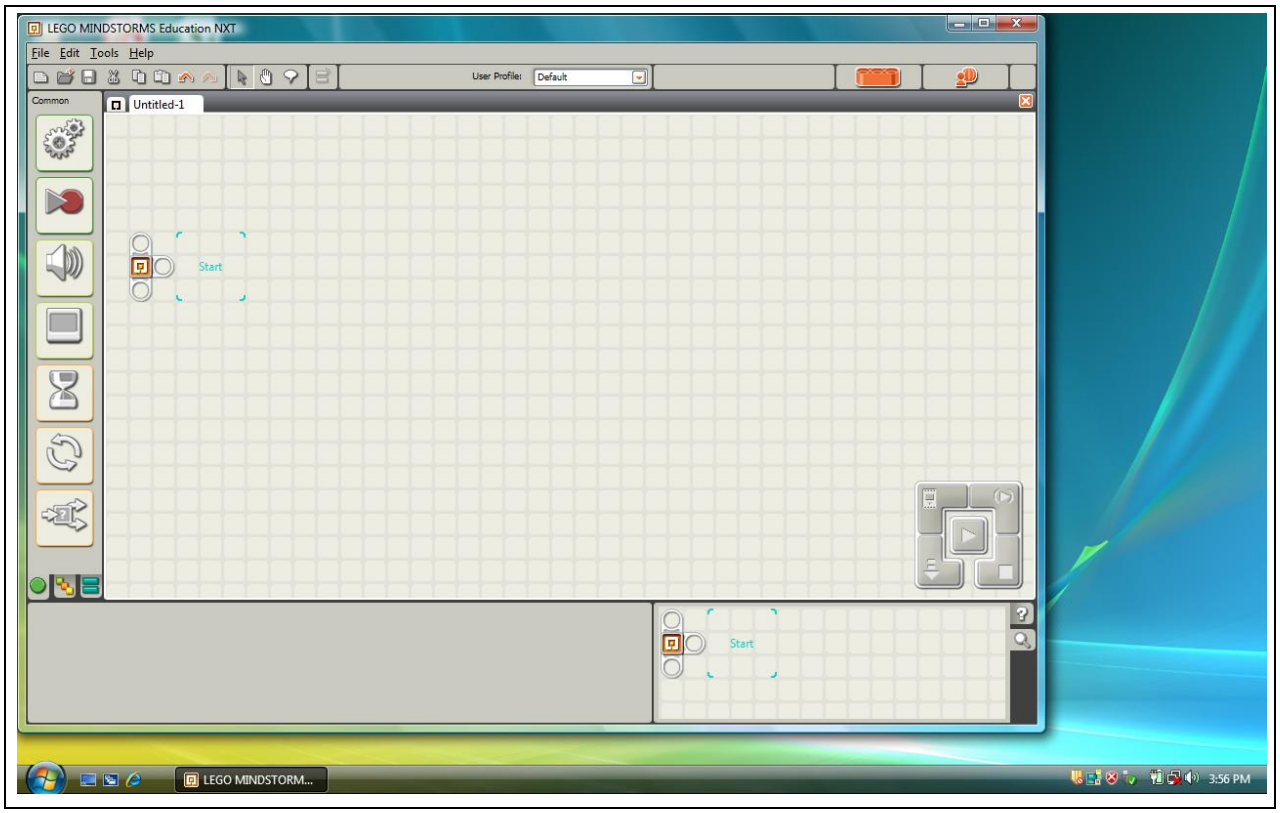


Figure C-1. The graphical programming interface.

Next, we demonstrate building a program by selecting the motor block at top left, dragging it to the central space labeled start, and dropping it. The result is shown in figure C-2, below.

The program icon now where the start block was represents a motor block, in particular, a motor block that controls the C and B motors, which in our case control the left and right wheels of the *Explorer*.

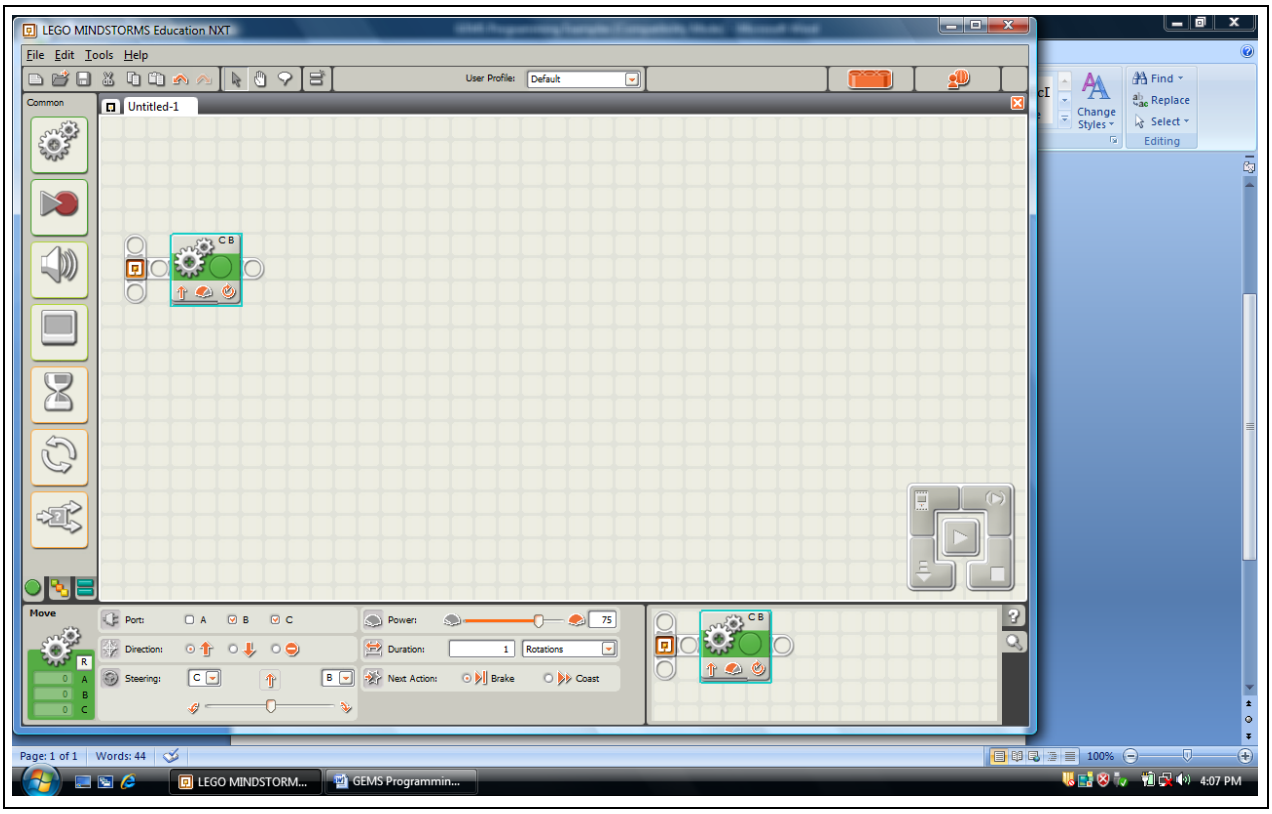


Figure C-2. A program consisting of one motor block.

Notice that the rectangular area at lower left and center, formerly blank, is now populated with some radio buttons, a slide control, and some other opportunities for input. These are the parameters that control the motors.

The port buttons labeled A, B, and C specify which ports are controlled by this particular motor block, in this case, the C and B motors which drive the wheels of the *Explorer*. Below them are three radio buttons for selecting forward movement, backward movement, or stopping. A slide control for steering occupies the bottom place in that column and controls how movement will be apportioned between the two motors. Another slide control for power tops the central column in the lower portion of the screen; below it is a duration control, which can be specified in terms of time, number of rotations, or angular degrees of rotation. The lowest control in that central column allows one to select whether the motors will brake to a stop or coast after completion of the specified motion.

Notice that the central portion of the screen, there is a multi-part square shaped figure at lower right. The buttons of this control send commands to the robot, including commands to download the displayed program, to stop, to start and so on. Commands and data are transmitted either by means of a connected USB cable or by Bluetooth wireless.

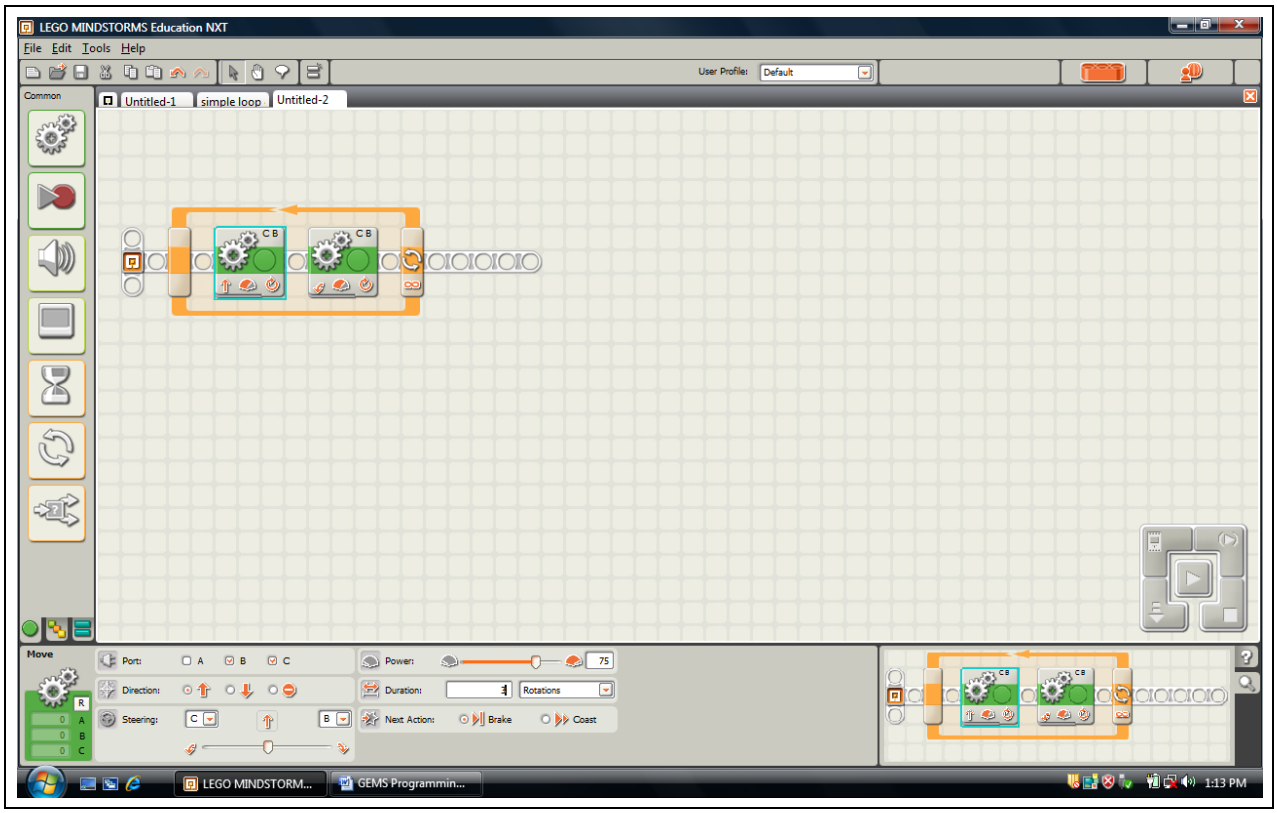


Figure C-3. A simple program incorporating a loop.

The program shown in figure C-3 incorporates a new construct, a loop, and two of the motor blocks which we previously encountered. When started, the program executes the motor blocks in succession, first moving forward for three rotations, then turning sharply for one rotation, and the repeats. In the figure, the first motor block is selected, as can be seen from the light blue square surrounding it, so the control parameter block at bottom left shows the parameters associated with it. If the outer loop had been selected, the loop control parameters would have been displayed. Since the loop counter is set to infinity, it will attempt to continue executing these until it is shut down manually.

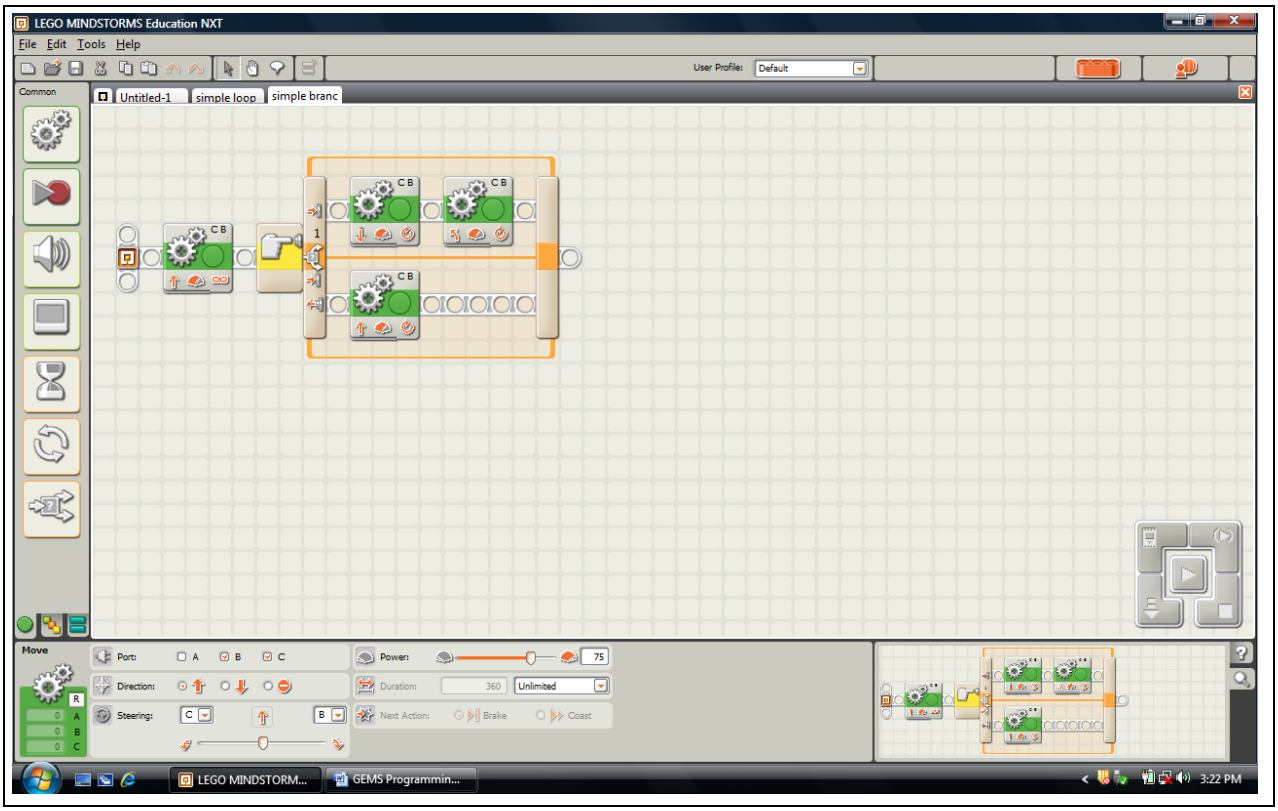


Figure C-4. A program including a switch controlled by a touch sensor.

Figure C-4 shows one more elaboration of the control language, a switch statement controlled by a touch sensor. Here the initial instruction is to move forward, but the larger yellow structure with a schematic finger pushing a button represents a switch controlled by a touch sensor. While the motor blocks are associated with output ports labeled by A, B, and C, the sensors are associated with input ports, numbered 1, 2, 3, and 4. Here the upper line in the branch block is executed when the touch sensor detects an obstacle, in this example causing the vehicle to first back up, then turn to the left, and then stop. The switch construct is essentially an "if, then, else" statement, with the upper branch representing the "if" part, and the lower part representing the "else" branch.

More complicated behaviors occur when movement, branching, and looping are all combined, as in the following program.

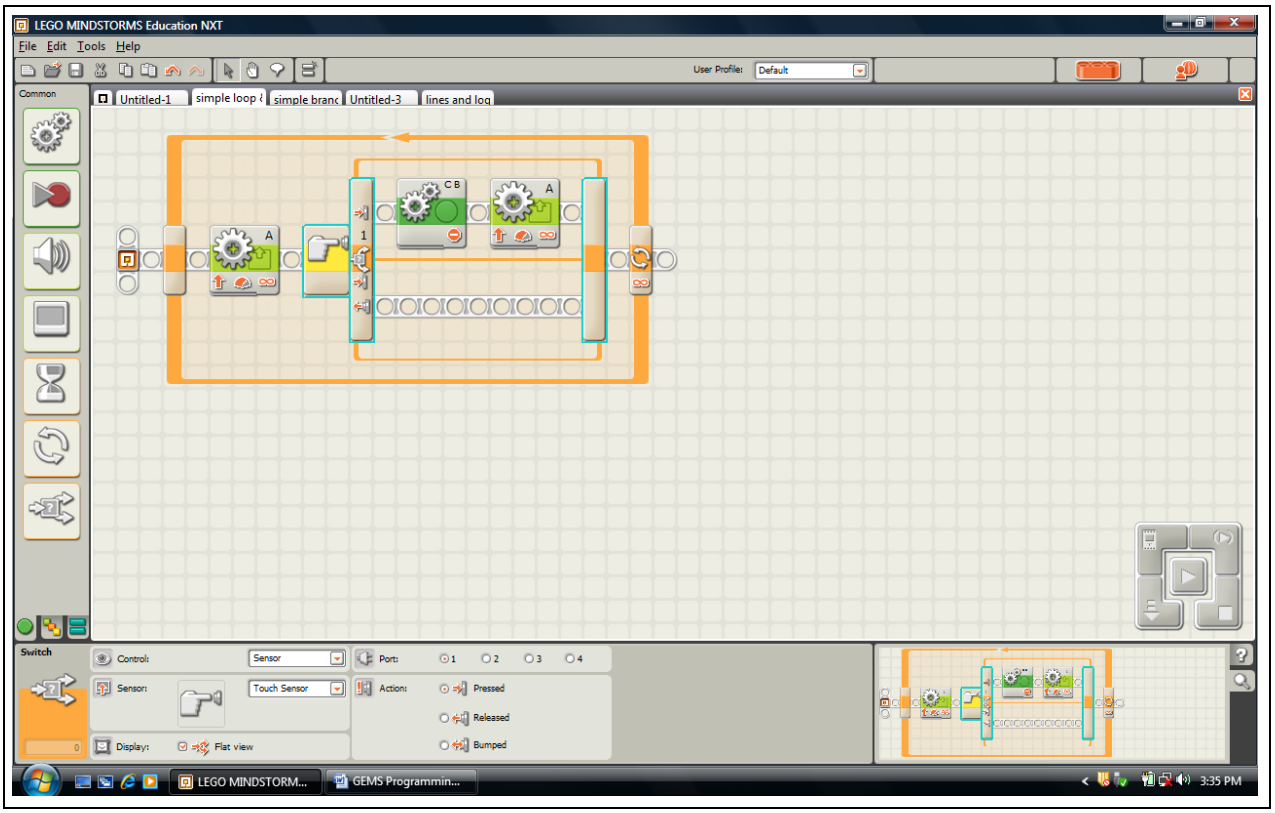


Figure C-5. A program with loop and switch blocks.

Here we see a slightly different version of the previous program enclosed now in a loop structure (figure C-5). In this case, the single line of motor blocks inside the touch controlled branching statement executes only if the touch sensor detects an obstacle. Because of the outer loop, the behavior of our bot is now to proceed until it detects an obstacle, back up, turn left, and then loop back to the start and proceed again.

The programming language has lots of other constructs, but for our purposes we will concentrate on just two more: data lines and signal processing blocks. Sensors can put out data signals, and the signal processing blocks can combine them and produce output data. That output data in turn can be sent to control structures like loops and branches.

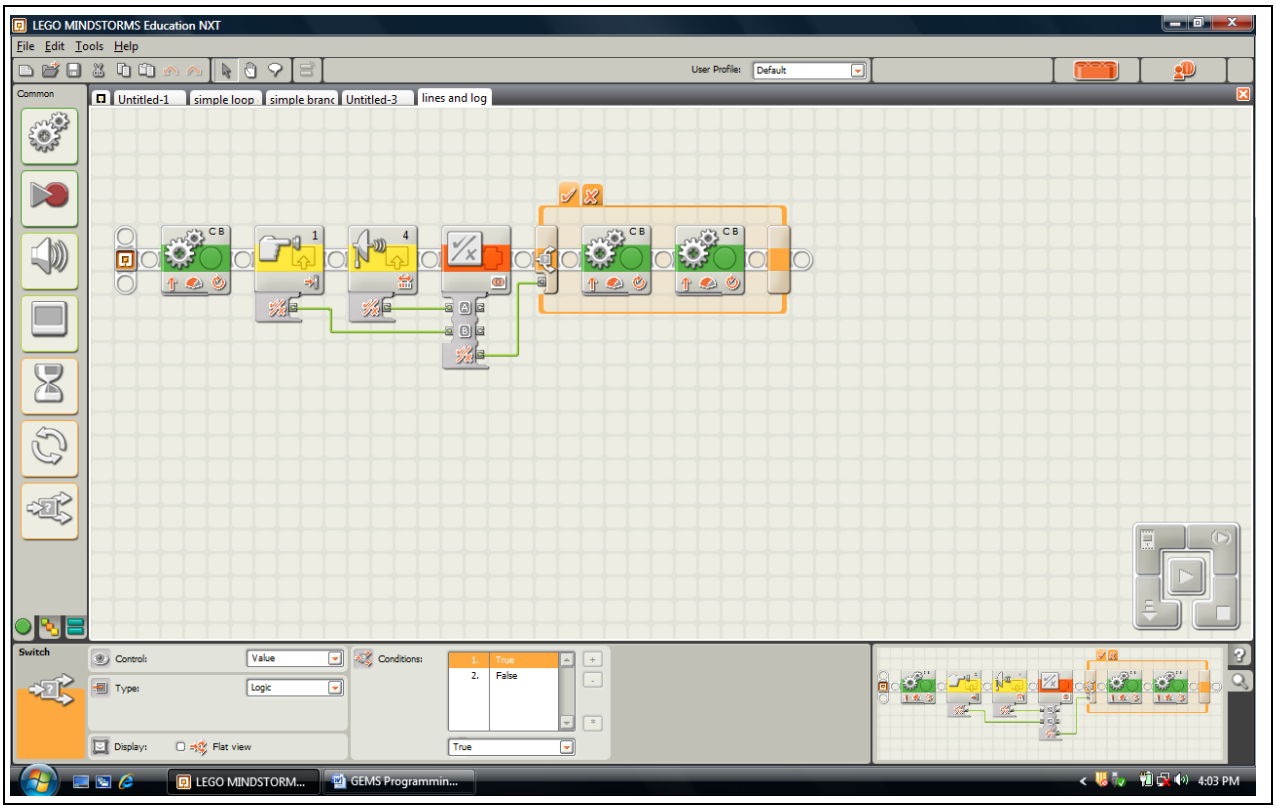


Figure C-6. A program with logic function and data lines.

Our latest program shows the new elements (figure C-6). After a motor block, we have two sensors in series, and each has a green line coming out the bottom and going to an orange block ahead, which in turn sends a green line on to the branch controller ahead. The two sensors are first a touch sensor and second an ultrasonic sensor. If either detects an obstacle within its range, it puts a true signal on the green line leading out of it. The orange block can be set up to be (for example) an "or" statement, in which case it puts a true on its green line that it sends to the branch block.

Consequently, if any sensor detects an obstacle, it puts out a true, which causes the logic block to put out a true, which causes the two motor blocks inside it to execute, causing the bot to back up, and turn. If we were to put the whole structure in a loop it would keep going, turning and backing and proceeding until stopped.

Once the students have experimented with these structures, they are ready to deal with the complexities of the full *Explorer* program.

When complete, the *Explorer* incorporates two drive wheel motors, powered by the B and C ports and has two sensors, a forward mounted touch sensor, and a steerable mast mounted ultrasonic sensor which can be pointed by the third motor (A port). When either sensor detects an obstacle, a sequence of events is triggered in which the system stops, determines whether it

touched something, in which case it utters an "Oof." (The controller units can produce various sound effects, a circumstance we prefer our students to discover later rather than sooner.) Next, the ultrasonic unit looks to each side, generating a distance measurement which it sends to a comparator, which information is used to control the robot to turn and proceed in the direction with the maximum unobstructed path. The program to control all that is shown in figure C-7.

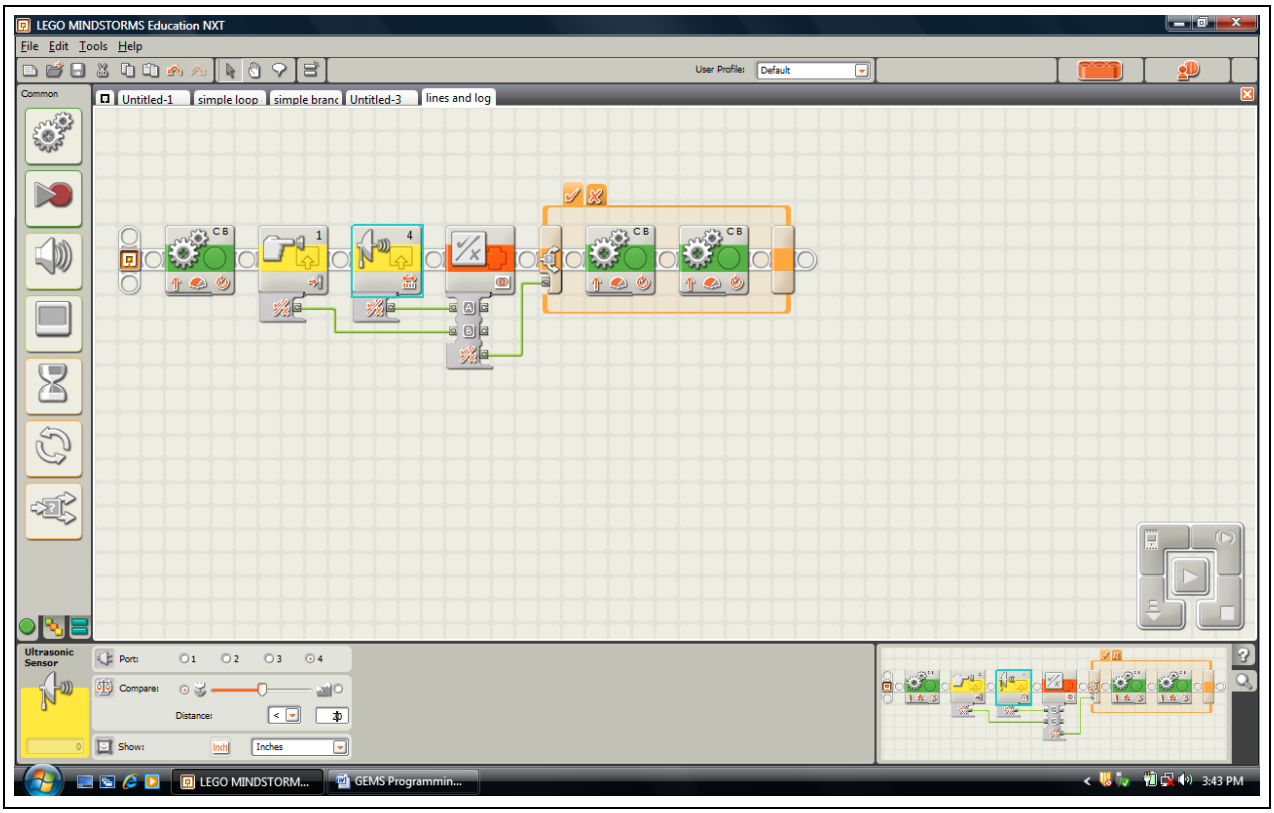


Figure C-7. Ultrasonic sensor control parameters in the lower left corner.

In the above view of the same program (figure C-7), the ultrasonic sensor block is selected, so the parameter block contains information associated with that sensor – the fact that it plugs into input port 4, that it is set to measure distance, and set to report true when the distance is less than 20 cm.

In C-8 below, the logic block is selected.

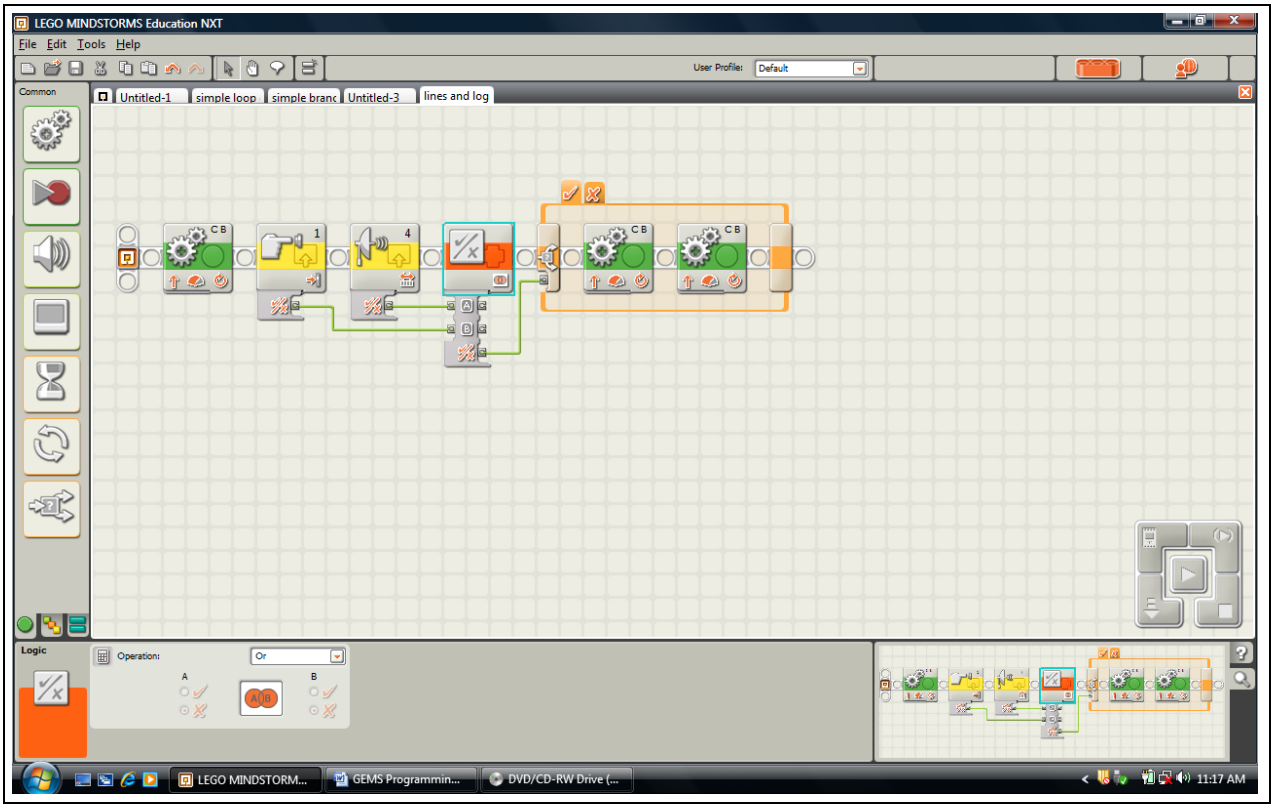


Figure C-8. This time, the logic block has been selected.

In figure C-9, the switch block is selected and its setting show that block execution is controlled by a logical value, that passed to it via the green line from the logic block and that the switch block executes on receiving a value of true.

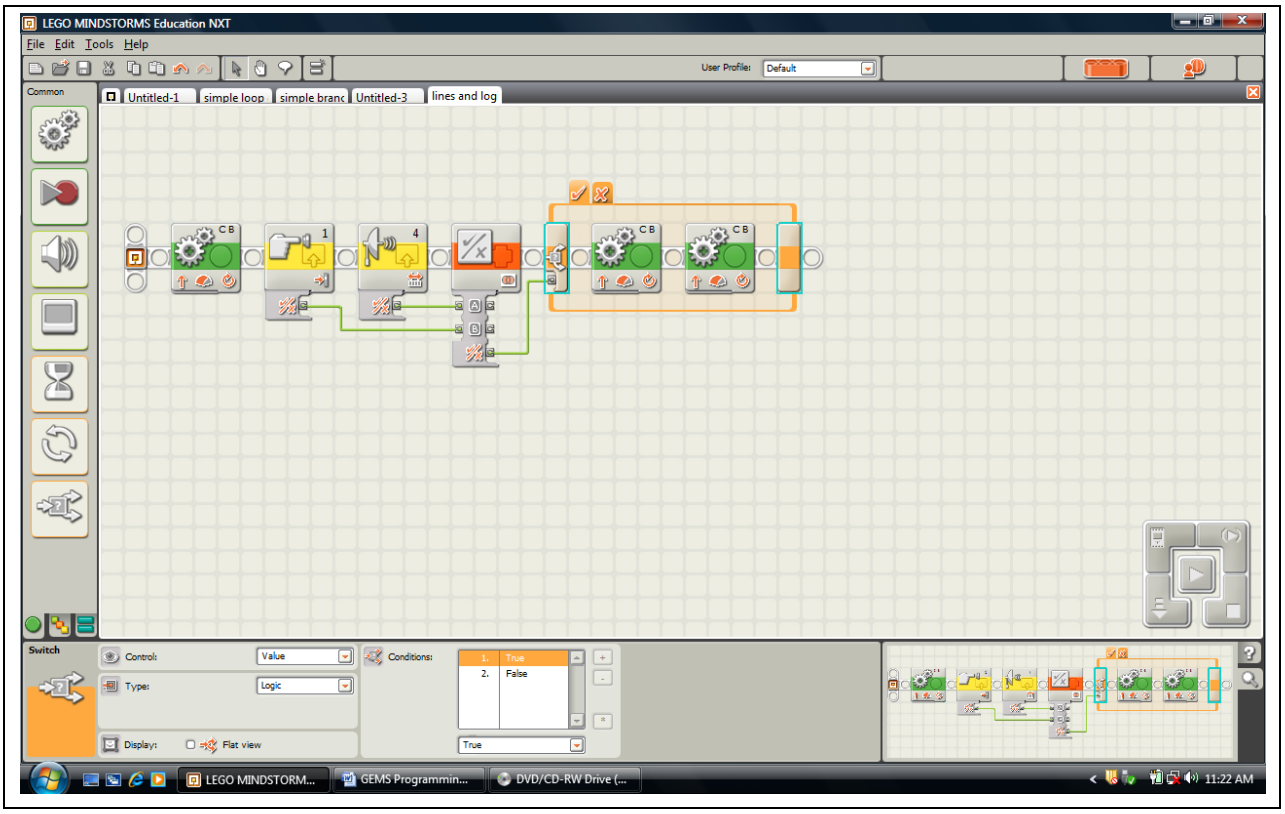


Figure C-9. Switch block using logic values.

The *Explorer* program is too big to fit conveniently on one page, so this is part one of two.

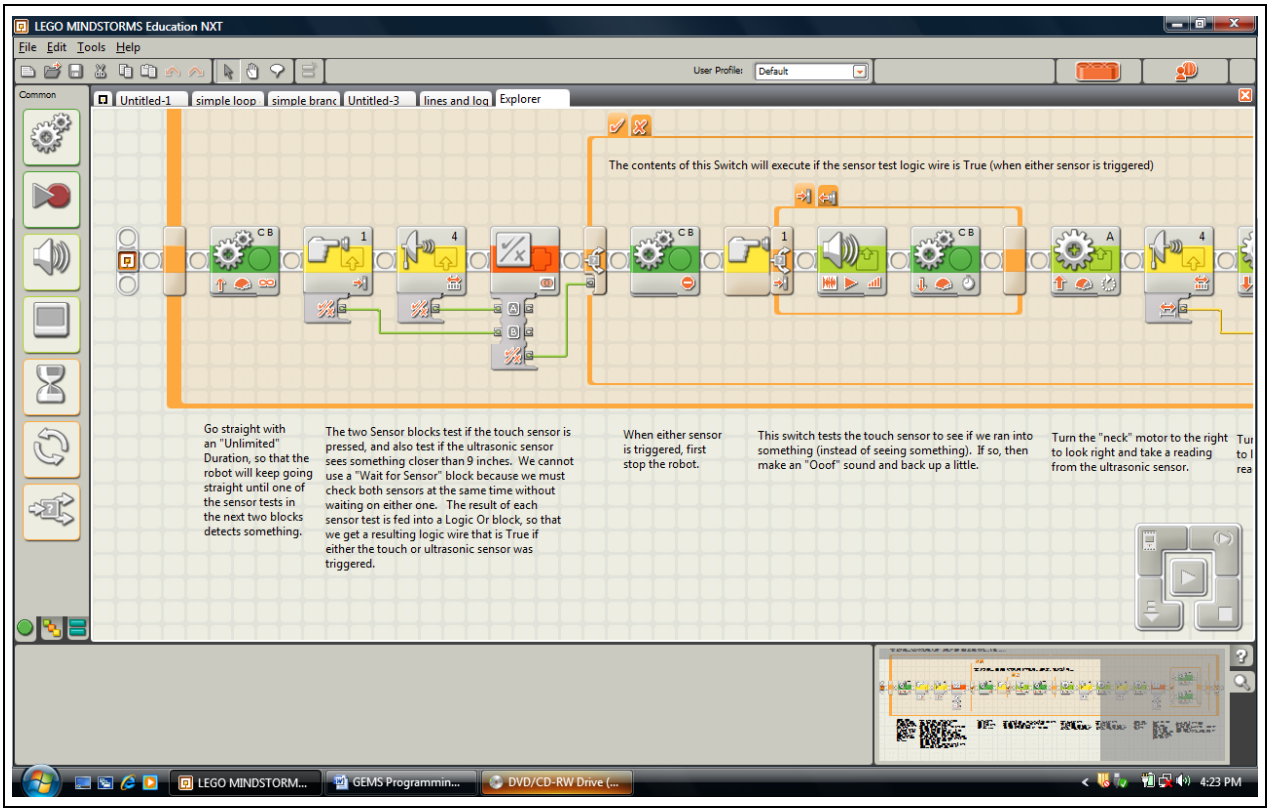


Figure C-10. *Explorer*, Part One See comments in the figure and below.

The initial block is an unlimited loop, so the program runs until it is manually stopped. The first five blocks inside the loop should look familiar, they are the same as the program we examined in the three previous figures. The green motor block sets the robot in motion and the touch sensor and the ultrasonic sensor (yellow blocks) output logical values of true if they detect an obstacle, which are combined in the red "or" block, which sends a true on to the yellow switch block if either of its inputs is true. When presented with a true, the program blocks inside the switch block execute.

If the switch block executes, the robot is stopped (first green block inside the switch block). Next the touch sensor is tested and an interior switch block executes if true. The interior switch block makes an "Oof" sound and backs the robot up a little. Discussion is found after the Part Two *Explorer* figure below.

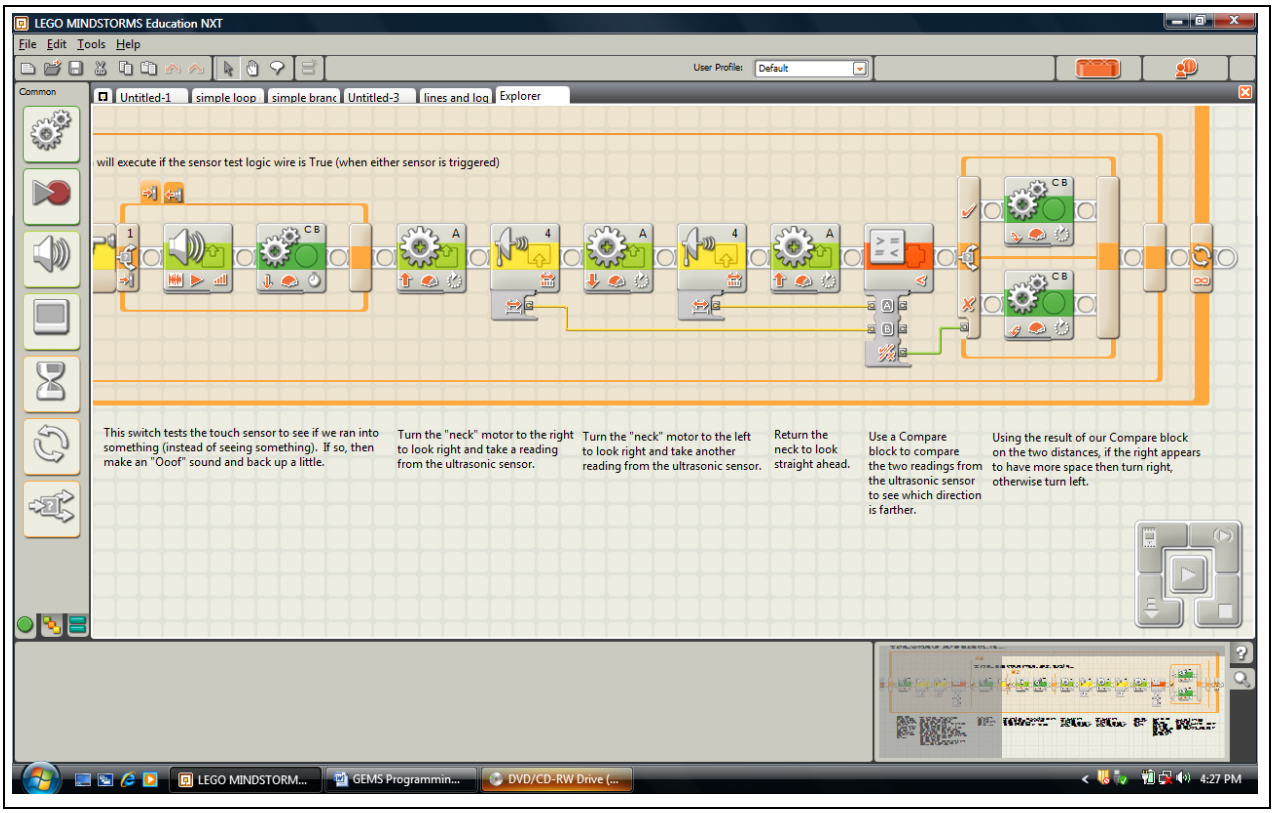


Figure C-11. *Explorer*, Part Two.

Note that we have already discussed the switch block on the left of this figure above, so our discussion will start with the green motor block labeled with an "A" in the upper right corner in contrast with our previous motor blocks labeled "CB." This label, and the single gear, indicates that the block controls only one motor and that that block is hooked up to the "A" port of the *Mindstorms* controller. What that motor does is control the direction in which the mast mounted ultrasonic sensor points. We won't show an explicit look into the parameter block of the "A" blocks, but what the first "A" block does is turn the ultrasonic sensor 90º to the left.

The following yellow block is the ultrasonic sensor and it outputs a value on the yellow line. Note the contrast with previous sensor blocks which output data on green lines. The difference is due to the fact that while those sensor blocks output logical (yes or no) values, this sensor block and the next put out numerical values, in this case the distance to the nearest obstacle in centimeters.

Next we have another "A" block which turns the ultrasonic sensor 180º to the right. Once again the sensor puts out a numerical value. The final "A" block turns the ultrasonic sensor 90º back to the left, restoring it to the forward facing direction.

After that, the outputs of the sensor are combined in a comparator block (red), which puts out a "true" logical value if the left side facing direction obstacle distance is greater than the right side facing obstacle distance. This value forms the input value of the following switch block. If true, the upper portion of the block executes and the *Explorer* robot turns to the left, otherwise, it turns to the right. Finally, the two interior switch blocks close and the outer loop block returns control to the start of the program.

To recapitulate the behavior of the *Explorer*, it starts moving and continues until it encounters an obstacle. If the touch sensor was activated, it says "Oof" and backs up. Then it checks to the left and the right and turns in the direction of longest clear path and proceeds as in the beginning. It can be quite amusing to watch it find its way through a maze of chair legs or human legs, but it can get stuck if it works itself into a corner from which its simple program can't extract itself.

NO OF COPIES	ORGANIZATION
1 PDF	DEFENSE TECH INFO CTR ATTN DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
2 HCS	US ARMY RSRCH LAB ATTN RDRL SLE E C RODRIGUEZ (2 COPIES) BLDG 1624 WHITE SANDS MISSILE RANGE NM 88002-5501
2 HCS	US ARMY RSRCH LAB ATTN RDRL VTU V C KRONINGER (2 COPIES) BLDG 1120A ABERDEEN PROVING GROUND MD 21005
2 HCS	US ARMY RSRCH LAB ATTN RDRL CIE D S. G. O'BRIEN ATTN RDRL CIE M T C JAMESON BATTLEFIELD ENVIR DIV BLDG 1622 WHITE SANDS MISSILE RANGE NM 88002-5001
1 HC 1 CD	US ARMY RSRCH LAB ATTN RDRL CIE D E MEASURE (1 HC 1 CD) BLDG 1622 WHITE SANDS MISSILE RANGE NM 88002-5501
2 HCS	US ARMY RSRCH LAB ATTN RDRL SLE A T MAXWELL (2 COPIES) BLDG 1624 WHITE SANDS MISSILE RANGE NM 88002-5513
2 HCS	US ARMY RSRCH LAB ATTN RDRL SLE G K AUSTIN (2 COPIES) BLDG 1622 WHITE SANDS MISSILE RANGE NM 88002-5513
5 HCS 1 CD	US ARMY RSRCH LABORATORY ATTN RDRL CIE D E CREEGAN (1 CD 5 HCS) BLDG 1622, ROOM 201 WHITE SANDS MISSILE RANGE NM 88002
5 HCS 1 CD	US ARMY RSRCH LAB ATTN IMAL HRA MAIL & RECORDS MGMT ATTN RDRL CIE BE DIVISION ATTN RDRL CIE D C KLIPP (1 CD, 1 HC) ATTN RDRL CIE S CHIEF ATTN RDRL CIO LL TECHL LIB ADELPHI MD 20783-1197